DOCUMENT RESUME

ED 183 144 IR 008 054

AUTHOR Moreau, Ken; Heines, Jessie M.
TITLE Creating Graphic Displays on Non-Graphics Terminals. Technical Report No. 5.
INSTITUTION Digital Equipment Corp., Maynard, Mass.
PUB DATE Apr 80
NOTE 9p.; Paper presented at the Annual Conference of the Association for the Development of Computer-Based Instructional Systems (Washington, DC, April 1980).

EDRS PRICE MF01/PC01 Plus Postage.
DESCRIPTORS *Computer Assisted Instruction; *Computer Graphics; *Production Techniques; *Programing; Visual Aids
IDENTIFIERS Computer Software

ABSTRACT

The interactive system for creating graphic displays that is described is presented as one solution to the problem created by the lack of complementary firmware or software to make the graphic capabilities of many current terminals easily accessible to the instructional programmer. The use of this system by the Digital Equipment Corporation's Computer-Based Course Develcpment Group for a computer-assisted instruction course that must run on two different types of video terminals with limited graphics capabilities (VT52 VT100) is discussed. (CMV)

****
* Reproductions supplied by EDRS are the best that can be made *
* from the original document. *
****

ED183144

U S DEPARTMENT OF HEALTH,
EDUCATION & WELFARE
NATIONAL INSTITUTE OF
EDUCATION

THIS DOCUMENT HAS BEEN REPRODUCED EXACTLY AS RECEIVED FROM THE PERSON OR ORGANIZATION ORIGINATING IT. POINTS OF VIEW OR OPINIONS STATED DO NOT NECESSARILY REPRESENT OFFICIAL NATIONAL INSTITUTE OF EDUCATION POSITION OR POLICY

CREATING GRAPHIC DISPLAYS ON NON-GRAPHICS TERMINALS

Ken Moreau
Jesse M. Heines

ABSTRACT

Graphic displays are an important tool for helping students visualize concepts presented by computer-based training materials. Many current terminals have limited graphic capabilities, but very few have complementary firmware or software to make those capabilities easily accessible to the instructional programmer. One solution to this problem is presented in this paper. An interactive system for creating graphic displays is described, and its use by Digital's Computer-Based Course Development Group is discussed.

"PERMISSION TO REPRODUCE THIS MATERIAL HAS BEEN GRANTED BY

Jesse M. Heines

Ken Moreau

TO THE EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)."

IR008054

CREATING GRAPHIC DISPLAYS ON NON-GRAPHICS TERMINALS

The Problem

Digital Equipment Corporation's Computer-Based Course Development Group is developing a computer-assisted instruction (CAI) course that must run on two different types of video terminals: the VT52, which is in wide use among DEC customers, and the newer VT100, which has more sophisticated display capabilities. Neither of these is a full "graphics terminal". However, the course designers wished to use some fairly extensive graphics, including direct cursor addressing, the drawing of boxes, blocks of text, animation of text, and complicated erasing and refreshing of selected portions of the screen.

Both of these terminals can perform the above actions by means of "escape sequences", which are streams of ASCII characters preceded by the ESCape character (decimal character number 27 in the ASCII character set). When the computer initiates an escape sequence, for instance, by executing a special PRINT statement in a BASIC program, the terminal will take the action specified rather than printing the characters in the escape sequence on the screen. For example, escape sequences can be used to move the cursor to a specified location, erase part or all of the screen, and print special graphics characters.

Every special action has a unique escape sequence, and these may vary from one terminal to another. On the VT52 for example, the escape sequence to place the cursor at the 10th row and 15th column is

(ESC)Y)-

where (ESC) denotes the escape character. On the VT100, which adheres to the ANSI standard escape sequences, the same result is accomplished by the escape sequence

(ESC)[10;15H

The VT100 also has capabilities that the VT52 lacks, such as bold and blinking characters, double height and double width lines, and reverse video.

Producing the instruction required on both terminals by coding the text and escape sequences into the instructional program is like trying to program in three languages simultaneously: the authoring language, VT52 escape sequences, and VT100 escape sequences. Also, CAI programs require huge amounts of text and graphics to explain and demonstrate the subject matter. The space that a single program can use is limited in any computer, but especially so on mini- and microcomputers. If highly graphic CAI programs are required to run on small machines, the instruc-

tion must be broken down into literally dozens of programs just to make them fit into the memory available. The control of these programs is complex, but more importantly from our point of view, passing control from one program to another takes an unacceptable period of time, leading to student dissatisfaction with the product.

Our Solution

We needed a tool that course developer could use to develop the displays needed to present a concept without requiring extensive training and the memorization of escape sequences. Our solution was to create a set of interpretive programs called DRAW. These programs incorporate high-level commands which are translated into the appropriate escape sequences. The commands are comparatively simple to remember, so the course developers can concentrate on the instructional value of various displays rather than the difficulty in programming them. Sample commands are shown in Table 1.

Since each terminal has different characteristics, the VT52 graphics editor simulates actions on the VT52 that may be primitives on the VT100 whenever possible. For example, the VT100 has an escape sequence to erase from the beginning of a line to the current cursor position. This primitive does not exist on the VT52, but the action is simulated by printing spaces from the beginning of the line to the current cursor position. The VT52 action is slower, but it produces the same result. The important point is that both the VT100 and the VT52 actions are initiated by a command of the form

```
.ebl (R,C)
```

which erases from row R column 1 to row R column C. Commands that are specific to the VT100 and that cannot be simulated, such as ".ron" to turn reverse video mode on, are simply ignored by the VT52 editor.

The DRAW approach to creating graphic displays has another advantage: the graphics information is stored separately from the CAI control programs. This allows the instructional programs to be smaller, and allows more program space to be devoted to complex answer judging and routing.

The Evolution of Graphics Displays

Creating graphics command source files. Graphic displays may be built with an interactive graphics editor, or files of graphic commands may be created with the normal system text editor. We have a different graphics editor for every terminal type that we use. All of these editors use the same command set, the same

Table 1

SAMPLE DRAW COMMANDS

Command	Action
.at (12,26)	position the cursor at line 12 column 26
.cen (24,40) DRAW	center the "DRAW" on line 24 around column 40
.box (10,12)(18,52)	draw a box with its upper left corner at line 10 column 12 and its lower right corner at line 18 column 52
.lin (5,4,35,r)	draw a line that extends from line 5 column 4, 35 spaces to the right
.ani "DRAW" (15,6,12,ur)	animate the word "DRAW" by moving it from row 15 column 6, 12 spaces up and to the right
.txb (8,20)	begin a text block at row 8 column 20 (the first text line that follows will begin at row 8 column 20, the second one at row 9 column 20, the third at row 10 column 20, etc., until another DRAW command is encountered)
.ron	turn reverse video mode on (all succeeding characters printed will be displayed as dark characters on a light background
.rof	turn reverse video mode off

syntax, and accept the same graphic command source files as input. The only difference (as far as the course developers are concerned) is the type of terminal on which the editor is being used.

The output of the graphics editors are graphics command source files. These files contain the source code of the DRAW system displays, and are completely compatible between all graphic editors. That is, any source file may be processed by any editor, regardless of whether it was created using that editor. As mentioned previously, some DRAW commands may be simulated or even ignored if they are not applicable to the terminal currently being used, but these commands will not generate errors and will not be deleted from the source code.

The source files consist of both DRAW commands and text. All DRAW commands start with a period as the first character on the line. All other lines are interpreted as text and are simply plotted on the screen at the current cursor position. Thus, a source file might consist of the following:

```
.box (3,15)(5,23)
.at (4,17) DRW52
.box (7,14)(9,23)
.at (8,16) DRW100
.ron
.txb (11,11)
 Interactive
    Graphics
     Editors
.rof
```

This source file would generate a display similar to that shown in Figure 1.

Source files are named according to the convention counnn.FRM. "cou" denotes a three-letter course code. For our project we have chosen the name EDT. "nnn" is the number of the display, from 000 to 999. ".FRM" is a fixed file extension used for all graphics command source files.

The graphics editors are highly interactive. When DRAW commands or text lines are entered, these actions cause immediate change

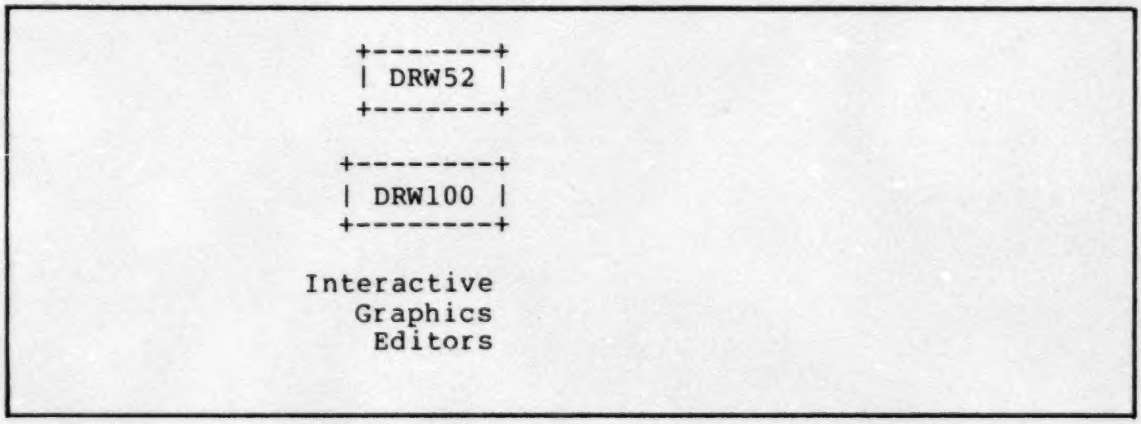


Figure 1

DISPLAY GENERATED FROM
SAMPLE GRAPHICS COMMAND SOURCE FILE

on the user's screen. The editors also provide three types of editing commands:

- character and line mode editing commands such as @INSERT, @REPLACE, and @SUBSTITUTE),
- file manipulation commands such as @OLD, @NEW, and @SAVE, and
- graphics manipulation commands such as @MOVE and @REPLOT.

The editing commands are differentiated from DRAW commands and text lines by the presence of an @ sign as their first character. Using the editing commands, graphics command source files can be read into memory, edited, executed, and saved as the course developer desires. The course developer can list and execute all or part of any display desired, edit the display, and then list or execute the result.

Translating graphics command source files. When the the final versions of the graphics command source files are created, they are submitted to the graphics translators. The translators compile the source files into ASCII streams which are stored in random access files on disk. Like the graphics editors, there is a translator program for each type of terminal.

The translators are very similar in principle to the graphics editors, but instead of presenting the effects of each command on the screen, the translators store them on disk for later access. The ASCII streams created by the translators contain all of the escape sequences necessary to perform the actions specified in the graphics command source files.

The translator programs produce two types of random access files:

- library files, which contain the ASCII streams that actually format the screen and display the graphic data, and
- key files, which provide numeric indices to the library files, indicating the starting and ending record numbers for each display.

There is one library file and one key file for each type of terminal in the system.

The relationships between all of the steps discussed thus far is shown in Figure 2.

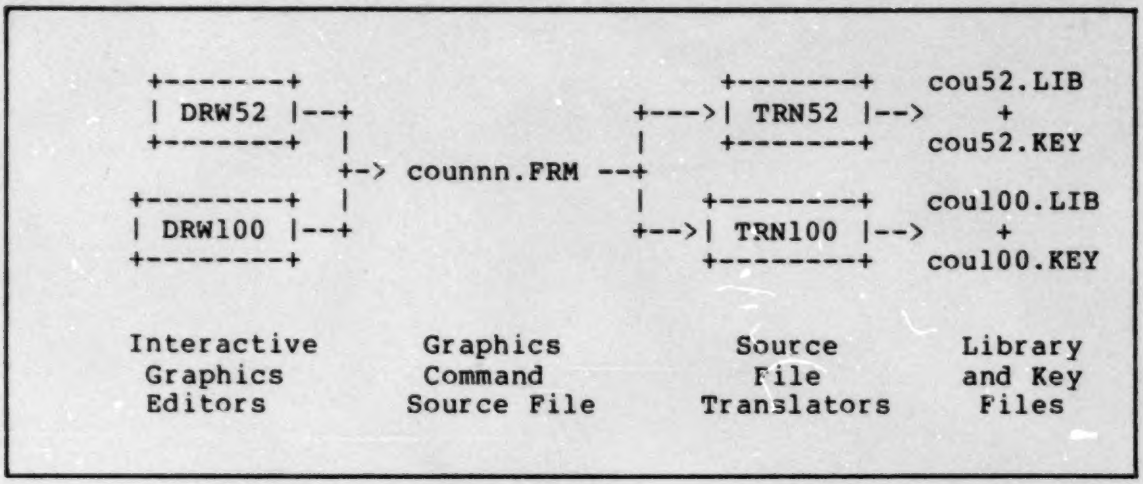


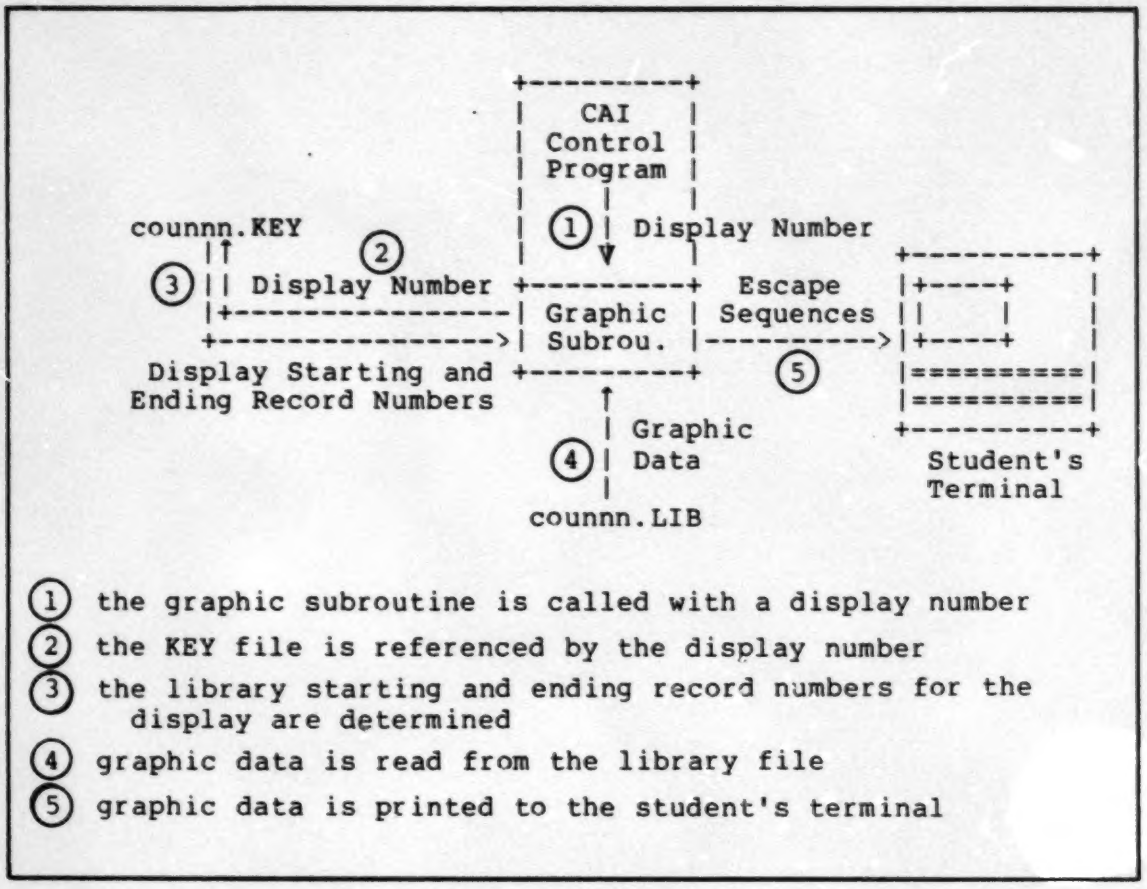
Figure 2

THE EVOLUTION OF GRAPHICS DISPLAYS

Calling Graphics Displays from Instructional Programs

The instructional control programs determine what type of terminal the student in running on at this moment, and access the corresponding random access files produced by the translators. In this way, the instructional lessons do not have to change for each terminal type on the system; they simply access the proper library and key files.

Presentation of each display is accomplished by a modular subprogram that is simply inserted into the instructional control program. The subroutine is called by the instructional lesson, with the number of a library display to be placed on the student's screen. This number corresponds to the number of a graphics command source file. For example, if the source file to be displayed is EDT123.FRM, the number passed to the graphics subroutine is 123. This calling number is used to reference the key file and look up the starting and ending record numbers of the display in the library file. The graphics subroutine then simply prints the appropriate records from the library file, and the display is shown to the student. This interaction is shown graphically in Figure 3.



1. the graphic subroutine is called with a display number
2. the KEY file is referenced by the display number
3. the library starting and ending record numbers for the display are determined
4. graphic data is read from the library file
5. graphic data is printed to the student's terminal

Figure 3

CALLING GRAPHICS DISPLAYS FROM INSTRUCTIONAL PROGRAMS

The DRAW System

The DRAW system is currently being used by Digital's Computer-Based Course Development Group to produce a CAI course on the DEC Standard Editor, EDT. We have found that using it enables the course developers to bring up complex displays quickly and easily, and enables the instructional programs to be devoted to more complex feedback and personalization of instruction.